Primal Wasserstein Imitation Learning

Robert Dadashi, Léonard Hussenot, Matthieu Geist, Olivier Pietquin

Primal Wasserstein Imitation Learning

Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin

Google Research, Brain Team

Abstract

Imitation Learning (IL) methods seek to match the behavior of an agent with that of an expert. In the present work, we propose a new IL method based on a conceptually simple algorithm: Primal Wasserstein Imitation Learning (PWIL), which ties to the primal form of the Wasserstein distance between the expert and the agent state-action distributions. We present a reward function which is derived offline, as opposed to recent adversarial IL algorithms that learn a reward function through interactions with the environment, and which requires little fine-tuning. We show that we can recover expert behavior on a variety of continuous control tasks of the MuJoCo domain in a sample efficient manner in terms of agent interactions and of expert interactions with the environment. Finally, we show that the behavior of the agent we train matches the behavior of the expert with the Wasserstein distance, rather than the commonly used proxy of performance.

1 Introduction

Reinforcement Learning (RL) has solved a number of difficult tasks whether in games [Tesauro, 1995, Mnih et al., 2015, Silver et al., 2016] or robotics [Abbeel and Ng, 2004, Andrychowicz et al., 2020]. However, RL relies on the existence of a reward function, that can be either hard to specify or too sparse to be used in practice. Imitation Learning (IL) is a paradigm that applies to these environments with *hard to specify rewards*: we seek to solve a task by learning a policy from a fixed number of demonstrations generated by an expert.

IL methods can typically be folded into two paradigms: Behavioral Cloning [Pomerleau, 1991, Bagnell et al., 2007, Ross and Bagnell, 2010] and Inverse Reinforcement Learning [Russell, 1998, Ng et al., 2000]. In Behavioral Cloning, we seek to recover the expert's behavior by directly learning a policy that *matches* the expert behavior in some sense. In Inverse Reinforcement Learning (IRL), we assume that the demonstrations come from an agent that acts optimally with respect to an unknown reward function that we seek to recover, to subsequently train an agent on it. Although IRL methods introduce an intermediary problem to solve (i.e. recovering the environment's reward) they offer the benefit of being less sensitive to distributional shift [Pomerleau, 1991], generalizing to environments with possibly changing dynamics [Piot et al., 2013], or recovering a near-optimal agent from suboptimal demonstrations [Brown et al., 2019, Jacq et al., 2019].

However, IRL methods are usually based on an iterative process alternating between reward estimation and RL, which might result in poor sample-efficiency. Earlier IRL methods [Ng et al., 2000, Abbeel and Ng, 2004, Ziebart et al., 2008] require multiple calls to a Markov decision process solver [Puterman, 2014], whereas recent adversarial IL approaches [Finn et al., 2016, Ho and Ermon, 2016, Fu et al., 2018] interleave the learning of the reward function with the learning process of the agent. Adversarial IL methods are based on an adversarial training paradigm similar to Generative Adversarial Networks (GANs) [Goodfellow et al., 2014], where the learned reward function can be thought of as the confusion of a discriminator that learns to differentiate expert transitions from non expert ones. These methods are well suited to the IL problem since they implicitely minimize an f-divergence between the state-action distribution of an expert and the state-action distribution of the

* Correspondence to Robert Dadashi: dadashi@google.com.

learning agent [Ghasemipour et al., 2019, Ke et al., 2019]. However the interaction between a generator (the policy) and the discriminator (the reward function) makes it a minmax optimization problem, and therefore comes with practical challenges that might include training instability, sensitivity to hyperparameters and poor sample efficiency.

In this work we use the Wasserstein distance as a measure between the state-action distributions of the expert and of the agent. Our approach is novel in the fact that we consider the problem of minimizing the Wasserstein distance through its primal formulation. We introduce an upper bound to the Wasserstein distance, which we verify empirically to be tight, and infer a conceptually simple reward function from it. The inferred reward function is non-stationary similarly to adversarial IL, but it is not re-evaluated as the agent interacts with the environment, therefore the reward function we define is computed *offline*. Crucially, this leads to a problem which is not a minmax optimization problem, and requires little fine tuning.

We make the following contributions. We introduce a reward function computed offline based on an upper bound of the primal form of the Wasserstein distance that does not need to be re-computed during the training procedure. We present a true distance to compare the behavior of the expert and the behavior of the agent, rather than using the common proxy of performance with respect to the true return of the task we consider (which is unknown in general). The method we present is conceptually simple and we show that it is able to recover expert behavior in MuJoCo environments with sample efficiency comparable to state-of-the-art methods both in terms of agent interactions with the environments and in terms of expert interactions with the environments.

2 Background and Notations

Markov decision processes We describe environments as episodic Markov Decision Processes (MDP) with finite time horizon [Sutton and Barto, 2018] $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \rho_0, T)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition kernel, r is the reward function, γ is the discount factor, ρ_0 is the initial state distribution and T is the time horizon. We will denote the dimensionality of $\mathcal{S}$ and $\mathcal{A}$ as $|\mathcal{S}|$ and $|\mathcal{A}|$ respectively. A policy π is a mapping from states to distributions over actions; we denote the space of all policies by Π. In RL, the goal is to learn a policy π that maximizes the expected sum of discounted rewards it encounters, that is, the expected return. Depending on the context, we might use the concept of a cost c rather than a reward r [Puterman, 2014], which essentially moves the goal of the policy from maximizing to minimizing its return.

State action distributions Suppose a policy π visits the successive states and actions $s_1, a_1, \dots, s_T, a_T$ during an episode, we define the empirical state-action distribution $\hat{\rho}_\pi$ as:

$$\hat{\rho}_\pi = \frac{1}{T} \sum_{t=1}^{T} \delta_{s_t, a_t},$$

where δ_{s_t, a_t} is a Dirac distribution centered on (s_t, a_t). Similarly, suppose we have a set of expert demonstrations $\mathcal{D} = \{s^e, a^e\}$ of size D, then the associated empirical expert state-action distribution $\hat{\rho}_e$ is defined as:

$$\hat{\rho}_e = \frac{1}{D} \sum_{(s,a) \in \mathcal{D}} \delta_{s,a}.$$

Wasserstein distance Suppose we have the metric space (M, d) where M is a set and d is a metric on M. Suppose we have μ and ν two distributions on M with finite moments, the p-th order Wasserstein distance [Villani, 2008] is defined as:

$$\mathcal{W}_p^p(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu, \nu)} \int_{M \times M} d(x, y)^p d\gamma(x, y),$$

where $\Gamma(\mu, \nu)$ is the set of all couplings between μ and ν.

In the rest of the paper we only consider distributions with finite support. A coupling between two distributions of support cardinal T and D is a doubly stochastic matrix of size $T \times D$. We note Γ the set of all doubly stochastic matrices of size $T \times D$, that is:

$$\Gamma = \Big\{\gamma \in \mathbb{R}_+^{T \times D} \mid \forall j \in [1:D], \sum_{i'=1}^{T} \gamma[i', j] = \frac{1}{D}, \forall i \in [1:T], \sum_{j'=1}^{D} \gamma[i, j'] = \frac{1}{T}\Big\}.$$

Optimizing a Wasserstein distance between distributions of state-action pairs requires the definition of a metric in the space $(\mathcal{S}, \mathcal{A})$. Defining a metric in an MDP can be highly non trivial [Givan et al., 2003, Ferns et al., 2004]; we extend on this subject in the appendix. From now on, we will assume the existence of a metric $d : (\mathcal{S}, \mathcal{A}) \times (\mathcal{S}, \mathcal{A}) \mapsto \mathbb{R}^+$.

3 Method

In this section, we present the theoretical motivation of our approach, which is the minimization of the Wasserstein distance between the state-action distributions of the agent and the expert. We introduce a reward based on an upper bound of the Wasserstein distance inferred from a relaxation of the optimal coupling condition, and present the resulting algorithm: Primal Wasserstein Imitation Learning (PWIL).

3.1 Wasserstein distance minimization

Central to our approach is the minimization of the Wasserstein distance between the state-action distribution of the policy we seek to train $\hat{\rho}_\pi$ and the state-action distribution of the expert $\hat{\rho}_e$. In other words, we aim at optimizing the following problem:

$$\inf_{\pi \in \Pi} \mathcal{W}_2^2(\hat{\rho}_\pi, \hat{\rho}_e) = \inf_{\pi \in \Pi} \inf_{\gamma \in \Gamma} \sum_{i=1}^{T} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma[i, j]. \tag{1}$$

We can interpret the Wasserstein distance using the earth's movers analogy [Villani, 2008]. Consider that the state-action pairs of the expert are D *holes* of mass $\frac{1}{D}$ and that the state-action pairs of the policy are *piles of dirt* of mass $\frac{1}{T}$. A coupling γ is a transport strategy between the piles of dirt and the holes, where $\gamma[i, j]$ stands for how much of the pile of dirt i should be moved towards the hole j. The optimal coupling is the one that minimizes the distance that the earth mover travels to put all piles of dirt to holes. Note that to compute the optimal coupling, we need knowledge of the locations of all piles of dirt. In the context of RL, this means having access to the full trajectory generated by π.

From now on, we write γ_π^* as the optimal coupling for the policy π, that is:

$$\gamma_\pi^* = \arg\min_{\gamma \in \Gamma} \sum_{i=1}^{T} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma[i, j].$$

We have:

$$\inf_{\pi \in \Pi} \mathcal{W}_2^2(\hat{\rho}_\pi, \hat{\rho}_e) = \inf_{\pi \in \Pi} \sum_{i=1}^{T} \underbrace{\sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_\pi^*[i, j]}_{= c_{i,\pi}^*}. \tag{2}$$

In the Equation (2), we have introduced $c_{i,\pi}^*$, which we interpret as a cost to minimize using RL. As $c_{i,\pi}^*$ depends on the optimal coupling γ_π^*, we can only define $c_{i,\pi}^*$ at the very end of an episode. This can be problematic if an agent learns in an online manner or in large time-horizon tasks. That is why we introduce an upper bound to the Wasserstein distance that yields a cost we can compute online, based on a suboptimal coupling strategy.

3.2 Greedy coupling

In this section we introduce the greedy coupling $\gamma_\pi^g \in \Gamma$, defined recursively for $1 \leq i \leq T$ as:

$$\gamma_\pi^g[i,:] = \underset{\gamma[i,:]\in\Gamma_i}{\arg\min} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma[i,j] \tag{3}$$

$$\text{with } \Gamma_i = \left\{ \gamma[i,:] \in \mathbb{R}_+^D \;\middle|\; \sum_{j'=1}^{D} \gamma[i,j'] = \frac{1}{T}, \forall j' \in [1:D], \sum_{i'=1}^{i} \gamma[i',j'] \leq \frac{1}{D} \right\}$$

Similarly to Equation (1), Equation (3) can be interpreted using the earth mover's analogy. Contrary to Equation (1) where we assume knowledge of the positions of the T piles of dirts, we now consider that they appear sequentially, and that the earth's mover need to transport the new pile of dirt to holes *right when it appears*. To do so, we derive the distances to all holes and move the new pile of dirt to the closest remaining available holes, hence the *greedy* nature of it. In Equation (3), the constraint $\sum_{j'=1}^{D} \gamma[i,j'] = \frac{1}{T}$ means that all the dirt that appears at the i-th timestep needs to be moved, and the constraint $\forall j' \in [1:D], \sum_{i'=1}^{i} \gamma[i',j'] \leq \frac{1}{D}$ means that we cannot fill the holes more than their capacity $\frac{1}{D}$. We show the difference between the greedy coupling and the optimal coupling in Figure 1, and provide the pseudo-code to derive it in Algorithm 1.

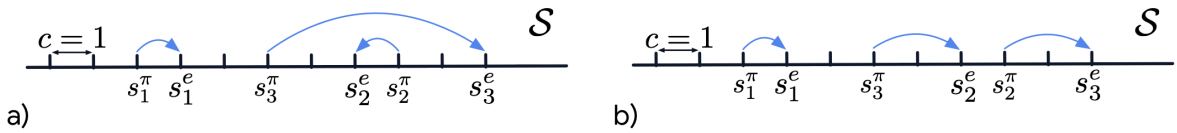

Figure 1: Illustration of the difference between the a) greedy coupling and the b) optimal coupling. We present an MDP where we drop the dependency on the action. The state space is $\mathbb{R}$ and the metric associated is the Euclidean distance. We note the states visited by the policy: $s_1^\pi, s_2^\pi, s_3^\pi$ and the states visited by the expert: s_1^e, s_2^e, s_3^e. When the policy encounters the state s_2^π, and because we do not know s_3^π yet, the greedy coupling strategy consists in coupling it with s_2^e although the optimal coupling strategy would be to couple it with s_3^e. Note that the total cost with the greedy coupling is 7 whereas the total cost with the optimal coupling is 5. This highlights that the optimal coupling needs knowledge about the whole policy's trajectory to be derived.

3.3 An upper bound to the Wasserstein distance

We can now define an upper bound of the Wasserstein distance using the greedy coupling (since by definition it is subotpimal):

$$\begin{aligned} \inf_{\pi\in\Pi} \mathcal{W}_2^2(\hat{\rho}_\pi, \hat{\rho}_e) &= \inf_{\pi\in\Pi} \sum_{i=1}^{T}\sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_\pi^*[i,j] \\ &\leq \inf_{\pi\in\Pi} \sum_{i=1}^{T}\sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_\pi^g[i,j] \end{aligned} \tag{4}$$

In Section 4, we empirically validate this upper bound. We can infer a cost from Equation (1) at each timestep i:

$$c_{i,\pi}^g = \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_\pi^g[i,j]. \tag{5}$$

Note that the greedy coupling $\gamma_\pi^g[t,.]$ defined in Equation (3) is dependent on all the previous state-actions visited by the policy π, which makes the cost $c_{i,\pi}^g$ non-stationary, similarly to adversarial IL

methods. Although this cost function is non-stationary, it does not have to be re-estimated as the agent interacts with the environment, and hence is said to be derived *offline*. We can infer a reward from the cost,

$$r_{i,\pi} = f(c^g_{i,\pi}),$$

where f is a monotonously decreasing function. Crucially, we have defined a reward function that we seek to maximize, without introducing an inner minimization problem (which is the case with adversarial IL approaches).

We can now derive an algorithm building on this reward function (Algorithm 1). The algorithm presented is written using pseudo-code for a generic agent A which implements a policy π_A, it observes tuples (s, a, r, s') and possibly updates its components. This formulation is general enough to show that our method is independent of the RL algorithm used. The algorithm computes the greedy coupling with a complexity $\mathcal{O}((|\mathcal{S}| + |\mathcal{A}|)D)$. Note that in the case where $T = D$, computing the greedy coupling is simply a lookup of the expert state-action pair that minimizes the distance with the agent state-action pair, followed by a pop-out of this minimizing expert state-action pair from the set of expert demonstrations.

Algorithm 1 PWIL: Primal Wasserstein Imitation Learning

Input: Expert demonstrations $\mathcal{D} = \{s^e_j, a^e_j\}_{j\in[1:D]}$, Agent A, number of episodes N
for $k = 1$ **to** N **do**
  Copy expert demonstrations with weight: $\mathcal{D}' := \{s^e_j, a^e_j, w^e_j\}_{j\in[1:D]}$, with $w^e_j = \frac{1}{D}$
  Reset environment, initial state s
  for $i = 1$ **to** T **do**
    Take action $a := \pi_A(s)$, observe next state s'
    Initialize weight $w^\pi := \frac{1}{T}$
    Initialise cost $c := 0$
    while $w^\pi > 0$ **do**
      Compute $s^e, a^e, w^e := \arg\min_{(s^e,a^e,w^e)\in\mathcal{D}'} d((s, a), (s^e, a^e))$
      if $w^\pi \geq w^e$ **then**
        $c := c + w^e d((s, a), (s^e, a^e))^2$
        $w^\pi := w^\pi - w^e$
        $\mathcal{D}'$.pop(s^e, a^e, w^e)
      else
        $c := c + w^\pi d((s, a), (s^e, a^e))^2$
        $w^e := w^e - w^\pi$
        $w^\pi := 0$
    $r := f(c)$
    A observes $(s,\ a,\ r,\ s')$
    A updates itself
    Update state $s := s'$

4 Experiments

In this section, we present the implementation of PWIL and perform an empirical evaluation that answers the following questions: 1) Can PWIL recover expert behavior? 2) How sample efficient is PWIL? 3) Does PWIL actually minimize the Wasserstein distance between the distributions of the agent and the expert? We also show the dependence of PWIL on multiple of its components through an ablation study. The complete description of the architecture of the agent and the hyperparameters search is provided in appendix; we will open-source the code in the near future.

4.1 Implementation

We test our method on five environments from the OpenAI Gym MuJoCo suite [Todorov et al., 2012, Brockman et al., 2016]: Walker2d-v2, Ant-v2, HalfCheetah-v2, Humanoid-v2, Hopper-v2. For each environment, multiple demonstrations are gathered using a DDPG agent with a distributional critic [Lillicrap et al., 2016, Barth-Maron et al., 2018] trained on the actual reward of these environments. Similarly to Ho and Ermon [2016], we subsample demonstrations by a factor of 20: we select one out of every 20 transitions with a random offset at the beginning of the episode. The subsampling of expert demonstrations makes the imitation task harder, so that a pure behavioral cloning approach does not perform to the level of the expert.

Central to our method is the existence of a metric between state-action pairs. We use the *standardized Euclidean distance* which is the L2 distance, weighted along each dimension by the inverse standard deviation of the expert demonstrations. From the cost c_i computed using Equation (5), we define a reward using the following:

$$r_i = \alpha \exp(-\frac{\beta T}{\sqrt{|\mathcal{S}| + |\mathcal{A}|}} c_i). \tag{6}$$

For all environments, we use $\alpha = 5$ and $\beta = 5$. We pick $f : x \mapsto \exp(-x)$ as the monotonously decreasing function to get a reward from a cost because it is smooth, and has a bounded range $[0, 1]$, which makes it well suited for a categorical distributional critic [Bellemare et al., 2017].

We test our method against a state-of-the-art imitation learning algorithm: Discriminator Actor Critic (DAC) [Kostrikov et al., 2019]. DAC is based on GAIL, and introduces an off-policy algorithm (TD3 [Fujimoto et al., 2018]) rather than an on-policy algorithm (TRPO [Schulman et al., 2015]). We use the open-source code provided by the authors. For our method, we use a DDPG agent [Lillicrap et al., 2016] with a distributional critic [Bellemare et al., 2017, Barth-Maron et al., 2018] (see details in the appendix). We initialize the replay buffer by filling it with expert state-action pairs with maximum reward (i.e. α).

We train PWIL and DAC in the limit of 1M environment interactions (2.5M for Humanoid) on 5 seeds. We run experiments with multiple numbers of demonstrations: 1, 4, 11 (consistently with previous work [Ho and Ermon, 2016, Kostrikov et al., 2019]). Every 10k environment steps, we perform 10 episodes rollouts per seed of the policy without exploration noise and report performance with respect to the environment's original reward in Figure 2. Learning curves on the inferred reward can be found in the appendix. We also provides visualisations of the final agents in `https://sites.google.com/view/wasserstein-imitation`.

4.2 Results

In Figure 2, PWIL shows consistent improvement on final performance for Hopper, Walker2d and Humanoid over DAC and similar performance for HalfCheetah. For Ant, PWIL outperforms DAC for 11 demonstrations and reaches similar performance for 1 and 4 demonstrations. Overall PWIL recovers expert behavior better than DAC over the considered environments at the 1M steps mark.

In terms of sample efficiency, DAC outperforms PWIL on HalfCheetah and Ant and has similar performance for Hopper and Walker2d. This seems rather contradictory with the claim that GAIL-like methods have poor sample efficiency. However, while DAC has state-of-the-art sample efficiency on all tasks but Humanoid, it requires careful tuning of the discriminator, including *e.g.*, a scheduled learning rate for the actor. We did not run a hyperparameter search to tune DAC on Humanoid, which might explain its poor performance on the task (results for Humanoid were not reported in [Kostrikov et al., 2019]). Nevertheless, this exemplifies the brittleness we mentioned for GAIL-like approaches.

Remarkably, PWIL is able to consistently learn policies on Humanoid with an average score over 6000 even with a single demonstration, which means that the agent we train can actually walk (some approaches in IL considers that Humanoid is solved when it can stand which corresponds to a score of 5000). Interestingly, on all environments but Humanoid, both PWIL and DAC are able to learn a policy with near-optimal expert performance with a single demonstration.

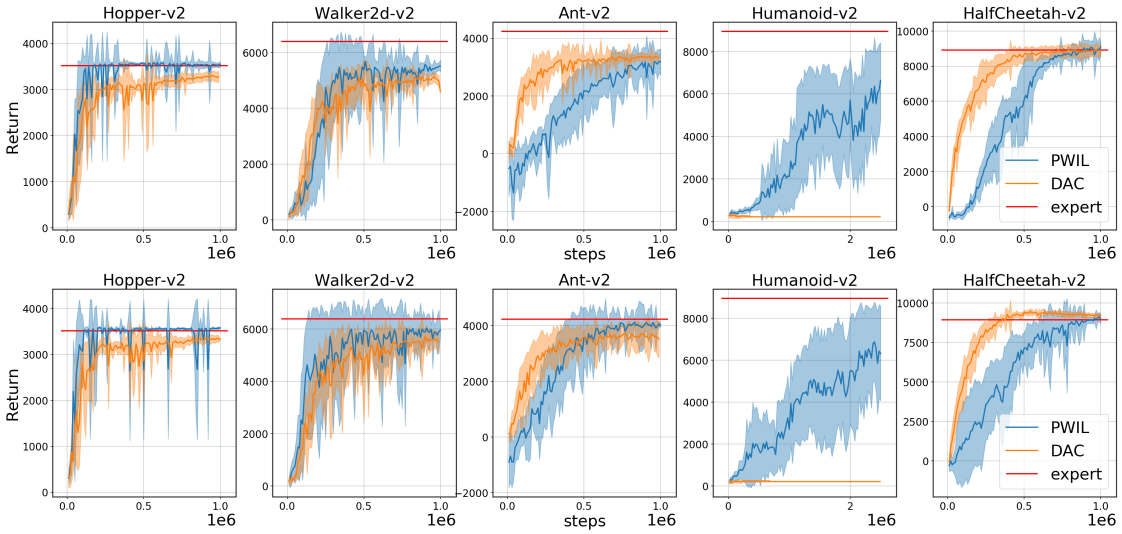

Figure 2: Mean and standard deviation return of the evaluation policy over 10 rollouts and 5 seeds, reported every 10k environment steps. The return here is in term of the environment's original reward. Top row: 1 demonstration, bottom row: 11 demonstrations.

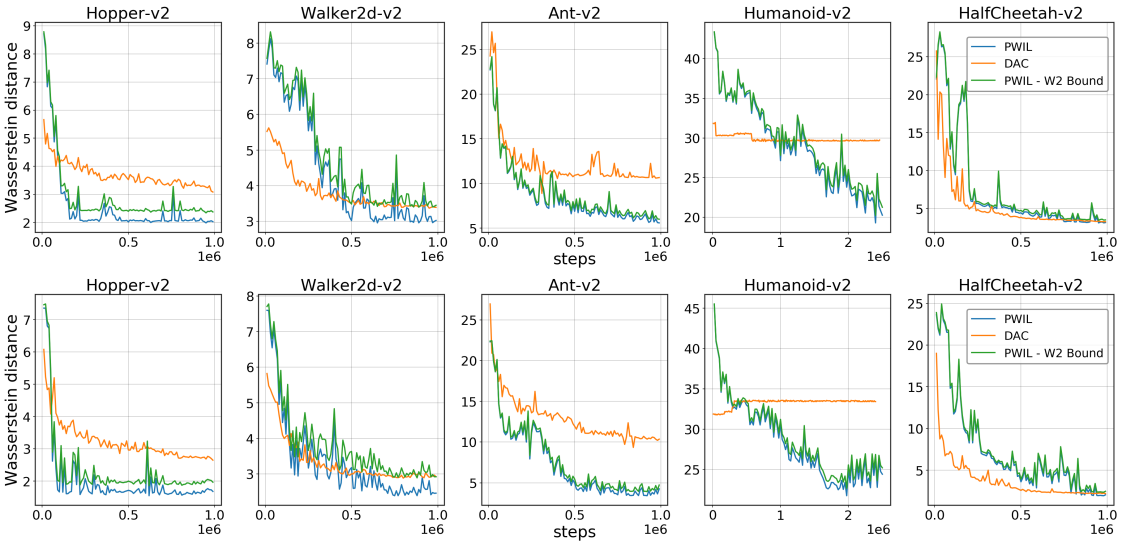

Figure 3: Mean of the Wasserstein distance between the state-action distribution of the evaluation policy and the state-action distribution of the expert over 10 rollouts and 5 seeds, reported every 10k environment steps. For PWIL, we include the upper bound on the Wasserstein distance based on the greedy coupling defined in Equation (4). Top row: 1 demonstration, bottom row: 11 demonstrations.

Wasserstein distance. In Figure 3, we show the Wasserstein distance to expert demonstrations throughout the learning procedure for PWIL and DAC. For both methods, we notice that the distance decreases while learning. PWIL leads to a smaller Wasserstein distance than DAC an all environments but HalfCheetah where it is similar.

PWIL defines a reward from an upper bound to the Wasserstein distance between the state-action distributions of the expert and the agent. We show in Figure 3 the Wasserstein distance as well as this upper bound throughout the learning procedure. Notice that our upper bound is "empirically tight", which validates the choice of the greedy coupling as an approximation to the optimal coupling.

We emphasize that this distance is useful in real settings of IL, regardless of the method used, *i.e.*

settings where we cannot have access to the actual reward of the task. Note that the convergence in terms of expected return (Figure 2) correlates with the convergence in terms of the Wasserstein distance (Figure 3), which opens interesting directions for e.g. early stopping of IL training procedures.

4.3 Ablation Study

In this section, we present the evaluation performance of PWIL over 5 seeds in the presence of ablations and report results in Table 1. We keep the hyperparameters of the original PWIL agent fixed. We provide the learning curves in the appendix.

	Hopper-v2	Walker2d-v2	Ant-v2	Humanoid-v2	HalfCheetah-v2
Expert	3548.0 ± 28.8	6131.7 ± 801.1	4177.0 ± 71.4	8966.3 ± 82.1	8879.1 ± 75.4
PWIL	3578.7 ± 14.3	5959.5 ± 142.5	4016.6 ± 69.3	6318.5 ± 2247.7	8975.4 ± 294.1
PWIL-state	3542.6 ± 55.7	3038.1 ± 1760.8	3277.2 ± 662.4	6640.8 ± 1960.2	9057.1 ± 727.8
PWIL-L2	3464.7 ± 46.2	4688.0 ± 2407.2	3212.2 ± 887.7	254.8 ± 95.3	2580.6 ± 2477.9
PWIL-nofill	3535.6 ± 70.7	5251.7 ± 677.7	3624.1 ± 151.9	5597.8 ± 2357.9	8828.8 ± 137.2
PWIL-support	3548.8 ± 40.6	3349.4 ± 1999.4	2626.0 ± 1175.9	7137.8 ± 1599.5	2285.2 ± 1260.1

Table 1: Ablation study of PWIL. Evaluation performance of variants of PWIL trained for 1M steps (2.5M for Humanoid) on 11 demonstrations. The numbers are the averages and standard deviations of the returns for 50 rollouts (10 rollouts per seed).

PWIL-state In this version of PWIL, we do not assume that we have access to the actions taken by the expert. Therefore, we try to match the state distribution of the agent with the state distribution of the expert (instead of the state-action distribution). The setup is referred to as Learning from Observation (LfO) [Torabi et al., 2018, Edwards et al., 2019]. The reward is defined similarly to PWIL, using a state distance rather than a state-action distance. Note that in this version, we cannot pre-fill the replay buffer with expert state-action pairs since we do not have access to actions. Remarkably, PWIL-state recovers non-trivial behaviors on all environments. We leave the extensive study of PWIL-state for future work.

PWIL-L2 In this version of PWIL, we use the Euclidean distance between state-action pairs, rather than the standardized Euclidean distance. In other words, we do not weight the state-action distance by the inverse standard deviation of the expert demonstrations along each dimension. There is a significant drop in performance for all environments but Hopper-v2. This shows that the performance of PWIL is sensitive to the quality of the MDP metric.

PWIL-nofill In this version, we do not prefill the replay buffer with expert transitions. This leads to a drop in performance which is significant for Walker and Ant. This should not come as a surprise since a number of IL methods leverage the idea of expert transitions in the replay buffer [Reddy et al., 2020, Hester et al., 2018].

PWIL-support In this version of PWIL, we define the reward as a function of the following cost:

$$\forall i \in [1:T],\ c_{i,\pi} = \inf_{\substack{\gamma_1,\dots,\gamma_D \in \mathbb{R} \\ \sum_{j=1}^{D} \gamma_j \leq \frac{1}{T}}} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^2 \gamma_j.$$

We can interpret this cost in the context of Section 3 as the problem of moving piles of dirt of mass $\frac{1}{T}$ into holes of infinite capacity. It is thus reminiscent of methods that consider IL as a support estimation problem [Wang et al., 2019, Brantley et al., 2020]. This leads to a significant drop in performance for Ant, Walker and HalfCheetah. Perhaps suprisingly, this boosts performance on Humanoid.

5 Related Work

Adversarial IL Similarly to our method, adversarial IL methods aim at matching the state-action distribution of the agent with the distribution of the expert, using different measures of similarity. For instance, GAIL [Ho and Ermon, 2016] considers the Shannon-Jensen divergence while AIRL [Fu et al., 2018] considers the Kullback-Leibler divergence. In the context of GANs, Arjovsky et al. [2017] show that replacing the f-divergence by the Wasserstein distance through the Kantorovich-Rubinstein duality [Villani, 2008] leads to better training stability, which a number of methods in IL have leveraged [Li et al., 2017, Lacotte et al., 2019, Kostrikov et al., 2019, Xiao et al., 2019, Liu et al., 2020]. However, the Kantorovich-Rubinstein duality only holds for the $\mathcal{W}_1$ distance [Peyré et al., 2019], its implementation comes with a number of practical approximations (*e.g.*, weight clipping to ensure Lipchitz continuity). Although these methods are based on the Wasserstein distance through its dual formulation, they drastically differ from ours as they rely on a minmax optimization problem. Recent approaches such as DAC [Kostrikov et al., 2019] demonstrate that adversarial IL approaches, while based on GAIL, can be sample efficient. However, they implicitly rely on a carefully tuned discriminator (e.g. network architecture, regularization strategy, scheduled learning rates, number of updates per environment interactions) that needs to interact well with a carefully tuned RL agent. This leads to potential brittleness, which is shown on Humanoid in Section 4. In contrast, the reward function we present relies on only two hyperparameters.

Expert support estimation Another line of research in IL consists in estimating the state-action support of the expert and define a reward that encourages the agent to stay on the support of the expert [Piot et al., 2014, Schroecker and Isbell, 2017, Wang et al., 2019, Brantley et al., 2020, Reddy et al., 2020]. Note that the agent might stay on the support of the expert without recovering its state-action distribution. Soft Q Imitation Learning [Reddy et al., 2020] assigns a reward of 1 to all expert transitions and 0 to all transitions the expert encounters; the method learns to recover expert behavior by balancing out the expert transitions with the agent transitions in the replay buffer of a value-based agent. Random Expert Distillation [Wang et al., 2019] estimates the support by using a neural network trained on expert transitions whose target is a fixed random neural network, similarly to [Burda et al., 2019]. Disagreement Regularized Imitation Learning [Brantley et al., 2020] estimates the expert support through the variance of an ensemble of BC agents, and use a reward based on the distance to the expert support and the KL divergence with the BC policies. Our method is similar in some sense to these methods since it is based on the distance to the support of the expert, with a support that *shrinks* through "pop-outs" to enforce the agent to visit the whole support. We showed in the ablation study that "pop-outs" are *sine qua non* for recovering expert behaviour (see *PWIL-support*).

6 Conclusion

In this work, we present Imitation Learning as a distribution matching problem and introduce a reward function which is based on an upper bound of the Wasserstein distance between the state-action distributions of the agent and the expert. The reward function we introduce is offline, in the sense that it does not need to be updated with interactions with the environment. It requires little tuning (2 hyperparameters) and it can recover near expert performance with as little as 1 demonstration on all considered environments, even the challenging Humanoid.

A number of IL methods are developed on synthetic tasks, where the evaluation of the IL method can be done with the actual return of the task. We emphasize that in our work, we present a true measure of similarity between the expert and the agent, that we can thus use in real IL settings, that is in settings where the reward of the task cannot be specified.

We think that our work opens multiple directions of research: the state-only version of PWIL learns non-trivial behavior in all tasks we considered, and could thus be extended to changing environment dynamics settings. Another interesting direction is the setting where a metric MDP cannot be specified as naturally as in the tasks we considered, for example in visual based observations.

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

Mohammed Amin Abdullah, Aldo Pacchiano, and Moez Draief. Reinforcement learning with wasserstein distance regularisation, with applications to multipolicy learning. *European Workshop on Reinforcement Learning*, 2018.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 2020.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

JA Bagnell, Joel Chestnutt, David M Bradley, and Nathan D Ratliff. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*, 2007.

Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *International Conference on Learning Representations*, 2018.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 2017.

Kianté Brantley, Wen Sun, and Mikael Henaff. Disagreement-regularized imitation learning. In *International Conference on Learning Representations*, 2020.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, 2019.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. *AAAI Conference on Artificial Intelligence*, 2020.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *International Conference on Learning Representations*, 2016.

Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *International Conference on Machine Learning*, 2019.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Conference on Uncertainty in Artificial Intelligence*, 2004.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations*, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, 2019.

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. *Conference on Robot Learning*, 2019.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 2003.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, 2018.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016.

Alexis Jacq, Matthieu Geist, Ana Paiva, and Olivier Pietquin. Learning from a learner. In *International Conference on Machine Learning*, 2019.

Liyiming Ke, Matt Barnes, Wen Sun, Gilwoo Lee, Sanjiban Choudhury, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *International Conference on Learning Representations*, 2019.

Jonathan Lacotte, Mohammad Ghavamzadeh, Yinlam Chow, and Marco Pavone. Risk-sensitive generative adversarial imitation learning. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2016.

Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. *International Conference on Learning Representations*, 2020.

Francisco S Melo and Manuel Lopes. Learning from demonstration using mdp induced metrics. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2010.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends in Machine Learning*, 2019.

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Learning from demonstrations: Is it worth estimating a reward function? In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted and reward-regularized classification for apprenticeship learning. In *International conference on autonomous agents and multi-agent systems*, 2014.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 1991.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: imitation learning via regularized behavioral cloning. *International Conference on Learning Representations*, 2020.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Stuart Russell. Learning agents for uncertain environments. In *Conference on Computational learning theory*, 1998.

Yannick Schroecker and Charles L Isbell. State aware imitation learning. In *Advances in Neural Information Processing Systems*, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 1995.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, 2012.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *International Joint Conference on Artificial Intelligence*, 2018.

Cédric Villani. *Optimal transport: old and new*. 2008.

Ruohan Wang, Carlo Ciliberto, Pierluigi Vito Amadori, and Yiannis Demiris. Random expert distillation: Imitation learning via expert policy support estimation. In *International Conference on Machine Learning*, 2019.

Huang Xiao, Michael Herman, Joerg Wagner, Sebastian Ziesche, Jalal Etesami, and Thai Hong Linh. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

A Agent Architecture

The agent we use is DDPG [Lillicrap et al., 2016] with a twin distributional critic [Fujimoto et al., 2018]. The actor architecture is a 4-layer neural network: the first layer has size 256 with tanh activation and layer normalization [Ba et al., 2016], the second layer and third layer have size 256 with elu activation [Clevert et al., 2016], the last layer is of size the dimension of the action space, with a tanh activation scaled to the action range of the environment. To enable sufficient exploration with use a Gaussian noise layer on top of the last layer with standard deviation $\sigma = 0.2$, that we clip to the action range of the environment. We evaluate the agent without exploration noise.

For the critic network we use a 4-layer neural network: the first layer has size 512 with tanh activation and layer normalization, the second layer is of size 512 with elu activation, the third layer is of size 256 with elu activation, the last layer is of dimension 101 with a softmax activation. The 101 neurons stand for the weights of the distribution supported within equal distance in the range $[-150, 150]$, as a categorical distribution [Bellemare et al., 2017].

We use the Adam optimizer [Kingma and Ba, 2015] with $\epsilon = 10^{-4}$ for both the critic and the actor. We use a batch size of 256. We clip both gradients from the critic and the actor to limit their L2 norm to 40.

We use a replay buffer of size 10^6, a discount factor $\gamma = 0.99$ and n step returns with $n = 5$. We prefill the replay buffer with $50k$ state-action pairs from the set of demonstrations (which means that we put multiple times the same expert transitions in the buffer).

We perform updates on the actor and the critic every $k = 4$ interactions with the environment. We did run a hyperparameter search on the following parameters:

Parameters	Values
ϵ	$10^{-5}, 10^{-4}, 10^{-3}$
σ	$0.1, 0.2, 0.3$
k	$2, 4, 8, 16$

Table 2: DDPG hyperparameters search.

For the reward function, we did run a hyperparameter search on α and β with the following values:

Parameters	Values
α	$1, 5, 10$
β	$1, 5, 10$

Table 3: Reward function hyperparameters search.

B Metrics in MDP

As PWIL is based on the Wasserstein distance, it relies on the existence of a metric in the MDP. The problem of designing a metric in an MDP remains a challenge. Bisimulation metrics [Ferns et al., 2004, Givan et al., 2003] define the concept of bisimilarity in MDPs via the following recurrent definition: two states are bisimilar if they yield the same expected reward and transition to bisimulation equivalence classes with equal probability. Recent work from Gelada et al. [2019] and Castro [2020] extend bisimulation metrics to the function approximation setting. Previous work has also assumed the existence of a MDP metric for imitation learning through kernel classification [Melo and Lopes, 2010] and for policy regularization using the Wasserstein distance [Abdullah et al., 2018]. We think that the problem of designing a metric and recovering the expert behavior through PWIL is an interesting direction for future work.

C PWIL Learning Curves

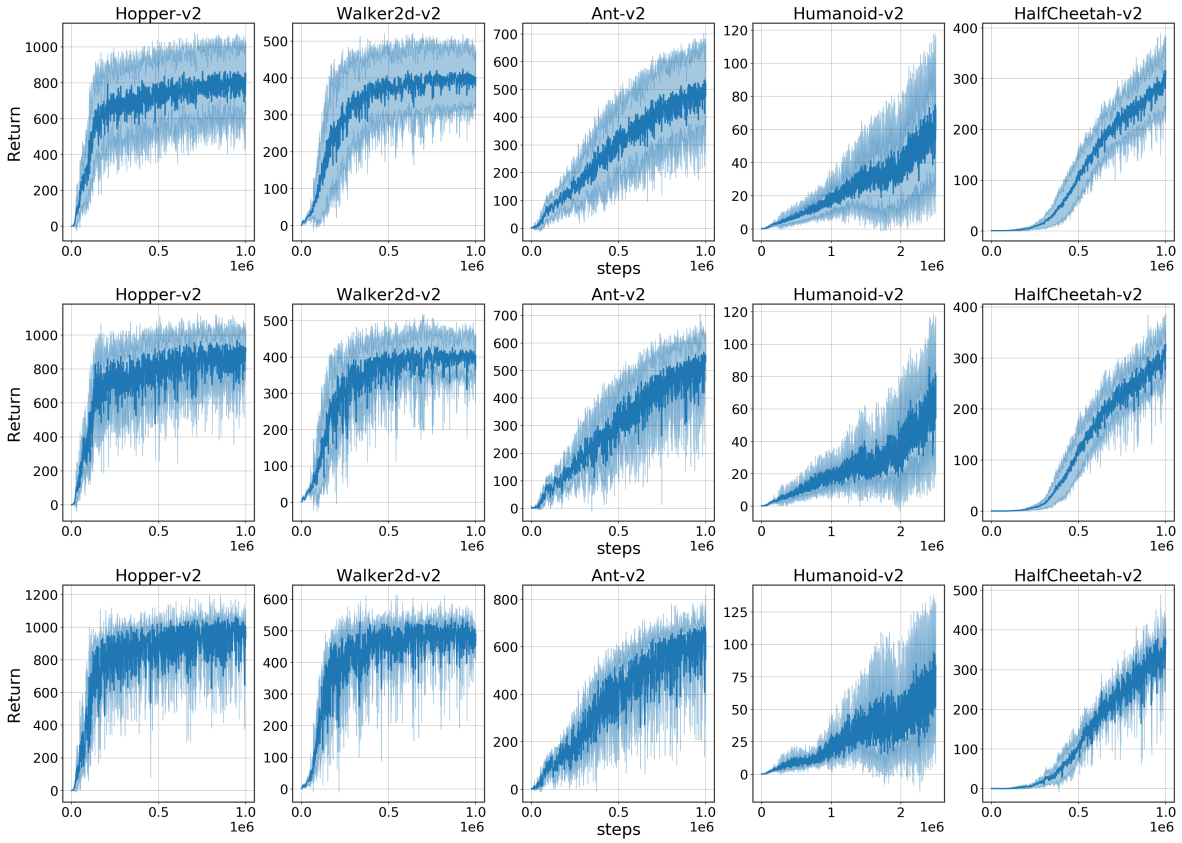

Figure 4: Mean and standard deviation of the return of the DDPG agent while learning (over the 5 seeds). The return here is in term of the reward defined with Equation (6). Top row: 1 demonstration, middle row: 4 demonstrations, bottom row 11 demonstrations.

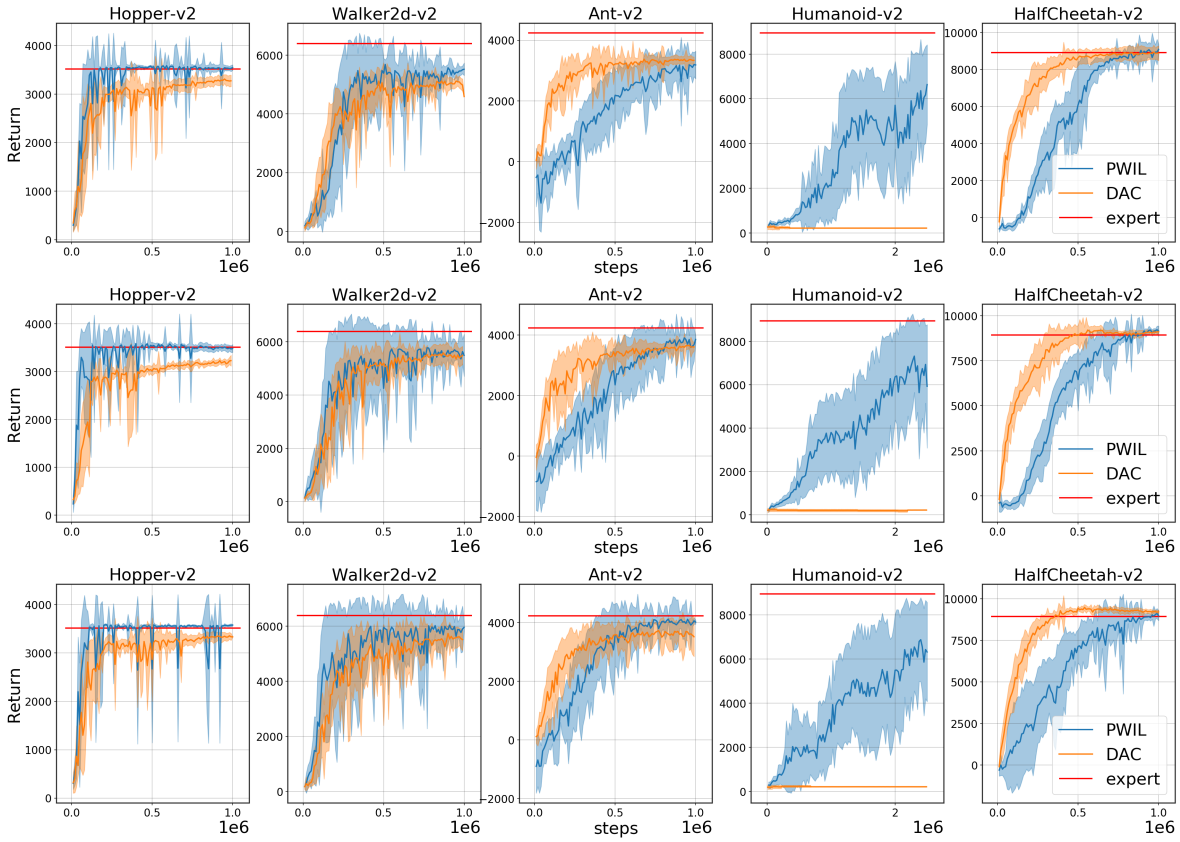

Figure 5: Mean and standard deviation of the original environment returns of the evaluation policy over 10 rollouts and 5 seeds, reported every 10k environment steps. Top row: 1 demonstation, middle row: 4 demonstrations, bottom row: 11 demonstrations.

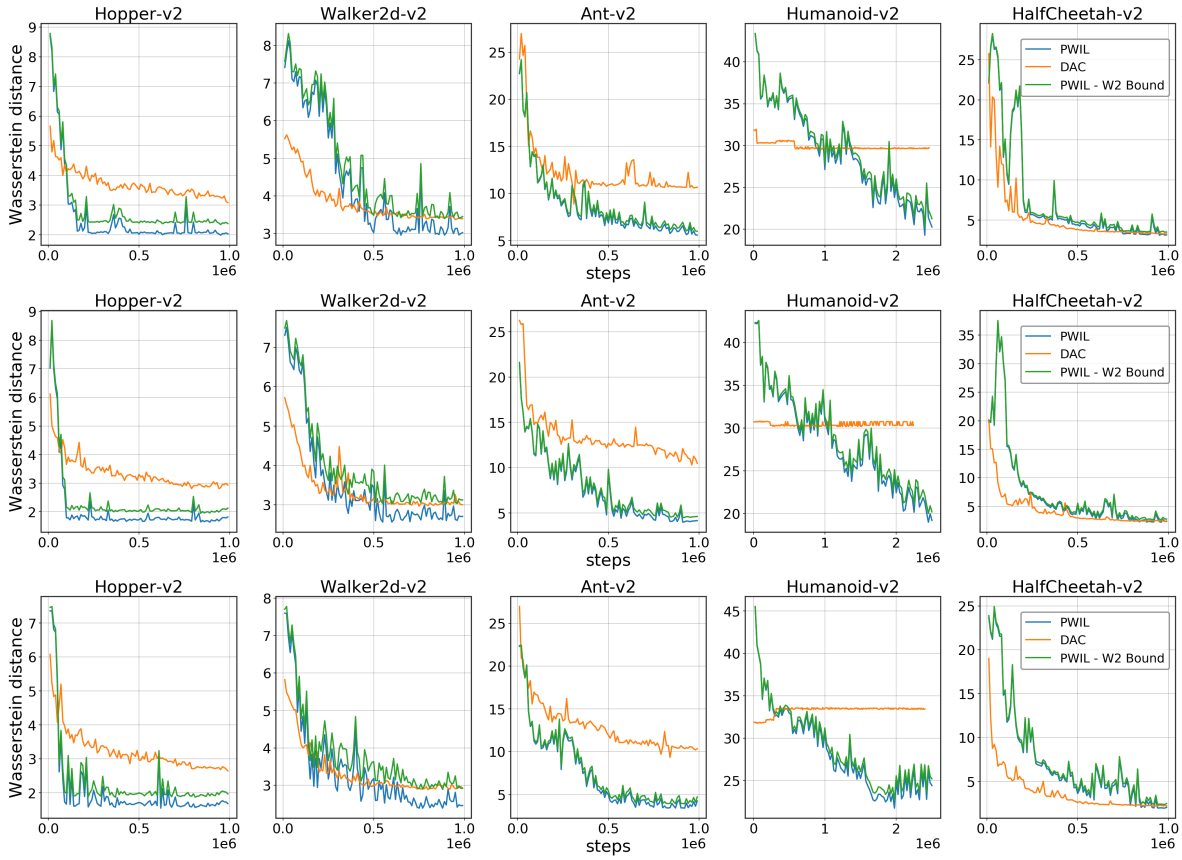

Figure 6: Mean of the Wasserstein distance between the state-action distribution of the evaluation policy and the state-action distribution of the expert over 10 rollouts and 5 seeds, reported every 10k environment steps. For PWIL, we include the upper bound on the Wasserstein distance based on the greedy coupling defined in Equation (4). Top row: 1 demonstration, middle row: 4 demonstrations, bottom row: 11 demonstrations.

D PWIL Ablation Study

	Hopper-v2	Walker2d-v2	Ant-v2	Humanoid-v2	HalfCheetah-v2
Expert	3548.0 ± 28.8	6131.7 ± 801.1	4177.0 ± 71.4	8966.3 ± 82.1	8879.1 ± 75.4
PWIL	3544.9 ± 56.4	5517.8 ± 213.3	3184.5 ± 455.1	6636.6 ± 1781.2	9089.5 ± 276.6
PWIL-state	3538.9 ± 33.4	3546.0 ± 1726.0	1780.6 ± 1042.1	5147.8 ± 2802.5	4984.0 ± 3764.7
PWIL-L2	3442.1 ± 49.2	4228.6 ± 1838.4	212.6 ± 1007.7	244.1 ± 108.5	-587.3 ± 249.6
PWIL-nofill	3469.2 ± 55.7	1886.1 ± 1804.9	3315.5 ± 203.6	5379.3 ± 2428.3	8678.8 ± 1355.8
PWIL-support	3557.7 ± 15.6	1232.5 ± 96.6	2268.6 ± 1056.1	6382.6 ± 1610.6	3401.5 ± 1370.7

Table 4: Ablation study of PWIL. Evaluation performance of variants of PWIL trained for 1M steps (2.5M for Humanoid) on 1 demonstrations. The numbers are the average and standard deviations of the return with the environment's original reward for 50 rollouts (10 rollouts per seed).

	Hopper-v2	Walker2d-v2	Ant-v2	Humanoid-v2	HalfCheetah-v2
Expert	3548.0 ± 28.8	6131.7 ± 801.1	4177.0 ± 71.4	8966.3 ± 82.1	8879.1 ± 75.4
PWIL	3486.3 ± 81.3	5490.0 ± 787.7	3856.5 ± 124.6	5924.2 ± 2844.7	9140.8 ± 287.6
PWIL-state	3552.1 ± 49.5	2583.6 ± 1929.7	2865.1 ± 372.5	4950.9 ± 2656.5	7007.5 ± 3619.0
PWIL-L2	3474.8 ± 16.4	5476.2 ± 1587.5	2457.6 ± 872.8	282.3 ± 101.8	-386.2 ± 564.8
PWIL-nofill	3469.2 ± 55.7	1886.1 ± 1804.9	3315.5 ± 203.6	5379.3 ± 2428.3	8678.8 ± 1355.8
PWIL-support	3548.8 ± 24.4	1370.1 ± 270.9	3053.6 ± 517.6	6354.3 ± 1669.2	2117.6 ± 642.2

Table 5: Ablation study of PWIL. Evaluation performance of variants of PWIL trained for 1M steps (2.5M for Humanoid) on 4 demonstrations. The numbers are the average and standard deviations of the return with the environment's original reward for 50 rollouts (10 rollouts per seed).

	Hopper-v2	Walker2d-v2	Ant-v2	Humanoid-v2	HalfCheetah-v2
Expert	3548.0 ± 28.8	6131.7 ± 801.1	4177.0 ± 71.4	8966.3 ± 82.1	8879.1 ± 75.4
PWIL	3578.7 ± 14.3	5959.5 ± 142.5	4016.6 ± 69.3	6318.5 ± 2247.7	8975.4 ± 294.1
PWIL-state	3542.6 ± 55.7	3038.1 ± 1760.8	3277.2 ± 662.4	6640.8 ± 1960.2	9057.1 ± 727.8
PWIL-L2	3464.7 ± 46.2	4688.0 ± 2407.2	3212.2 ± 887.7	254.8 ± 95.3	2580.6 ± 2477.9
PWIL-nofill	3535.6 ± 70.7	5251.7 ± 677.7	3624.1 ± 151.9	5597.8 ± 2357.9	8828.8 ± 137.2
PWIL-support	3548.8 ± 40.6	3349.4 ± 1999.4	2626.0 ± 1175.9	7137.8 ± 1599.5	2285.2 ± 1260.1

Table 6: Ablation study of PWIL. Evaluation performance of variants of PWIL trained for 1M steps (2.5M for Humanoid) on 11 demonstrations. The numbers are the average and standard deviations of the return with the environment's original reward for 50 rollouts (10 rollouts per seed).

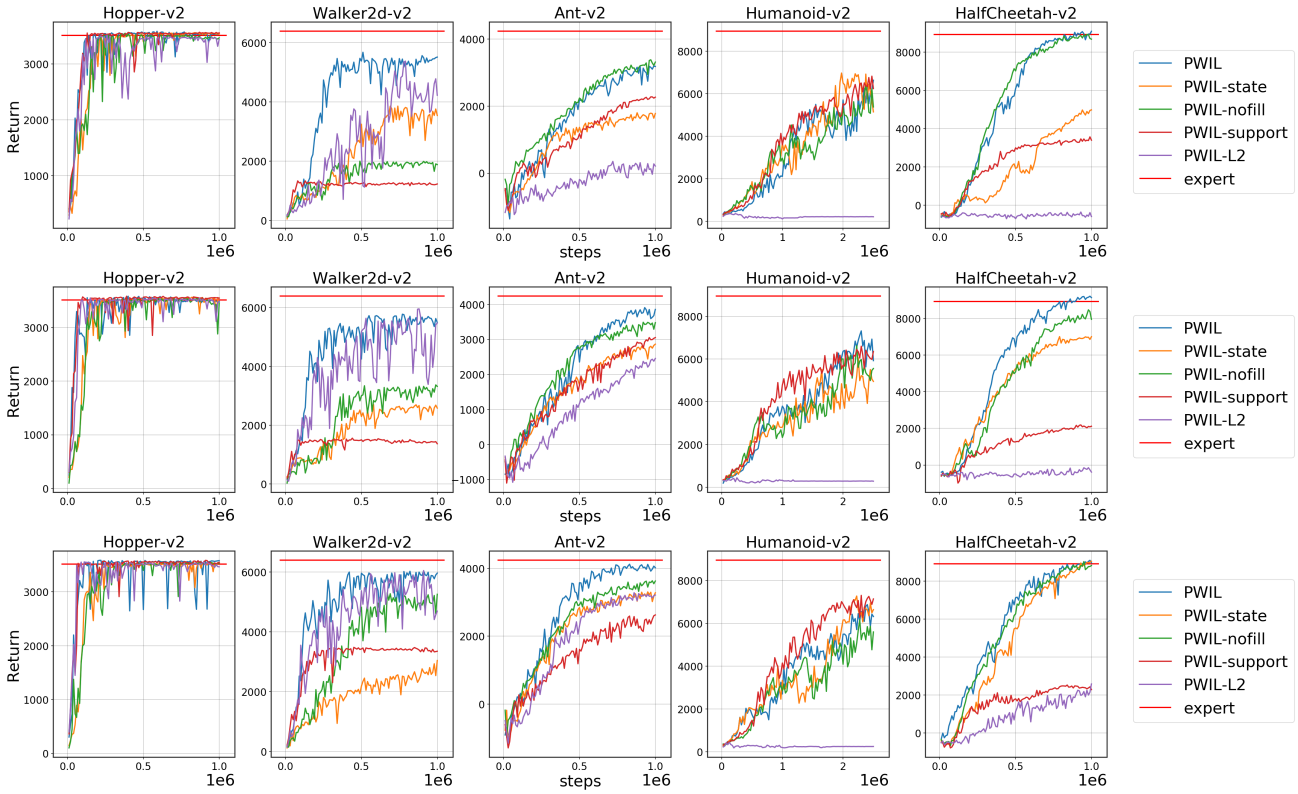

Figure 7: Mean of the evaluation performance of different variants of PWIL over 10 rollouts and 5 seeds, reported every 10k environment steps. The return here is in term of the environment's original reward. Top row: 1 demonstation, middle row: 4 demonstrations, bottom row: 11 demonstrations.